

## N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM  
MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT  
CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED  
IN THE INTEREST OF MAKING AVAILABLE AS MUCH  
INFORMATION AS POSSIBLE

# AgRISTARS

"Made available under NASA sponsorship  
in the interest of early and wide dis-  
semination of Earth Resources Survey  
Program information and without liability  
for any use made thereof."

80-10311  
GR-163379  
SR-P0-00469  
NAS9-15466

A Joint Program for  
Agriculture and  
Resources Inventory  
Surveys Through  
Aerospace  
Remote Sensing

## Supporting Research

July 1980

### Technical Report

## A Multispectral Data Simulation Technique

by Marwan J. Muasher and Philip H. Swain

(E80-10311) A MULTISPECTRAL DATA SIMULATION  
TECHNIQUE (Purdue Univ.) 27 p HC A03/MF A01  
CSCL 05B

N80-33830

Unclas

G3/43 00311

Purdue University  
Laboratory for Applications of Remote Sensing  
West Lafayette, Indiana 47907



NASA



SR-PO-00469

NAS9-15466

TECHNICAL REPORT

A MULTISPECTRAL DATA SIMULATION TECHNIQUE

By

M. J. Muasher

and

P. H. Swain

This report describes activity carried out  
in the Supporting Research Project.

PURDUE UNIVERSITY

LABORATORY FOR APPLICATIONS OF REMOTE SENSING

1220 Potter Drive

WEST LAFAYETTE, INDIANA 47906, U.S.A.

JULY 1980

# Star Information Form

1. Report No. SR-PO-00469		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle A MULTISPECTRAL DATA SIMULATION TECHNIQUE				5. Report Date July 1980	
				6. Performing Organization Code	
7. Author(s) Marwan C. Muasher and Philip H. Swain				8. Performing Organization Report No. 070980	
9. Performing Organization Name and Address Laboratory for Applications of Remote Sensing Purdue University West Lafayette, IN 47906				10. Work Unit No.	
				11. Contract or Grant No. NAS9-15466	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Johnson Space Center Houston, TX 77058 Tech. Monitor: Richard Heydorn				13. Type of Report and Period Covered	
				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract <p>In remote sensing data analysis, several assumptions are made that are not always precisely met. These assumptions include that the classes in the data are normally distributed, that training data are representative of the area of interest, that the number of classes is known, and that all pixels are pure.</p> <p>In testing new algorithms, deviations from the assumptions may obscure the action of the new process. One way to clarify the situation is to apply the algorithm first to a data set satisfying the assumptions.</p> <p>A method is presented to obtain an artificial data set through simulation. While retaining the natural spatial and spectral information in the scene by basing the simulation on a classification, the data set provides the analyst with an exact number of classes in the scene, true distributions of these classes, independent measurements and "pure" pixels.</p> <p>Program listings in both Fortran and C-Language are provided in the appendices.</p>					
17. Key Words (Suggested by Author(s)) Data Simulation Multispectral Data				18. Distribution Statement	
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages	
				22. Price*	

## A MULTISPECTRAL DATA SIMULATION TECHNIQUE\*

Marwan J. Muasher and Philip H. Swain

For remote sensing data analysis, several assumptions are commonly made. These assumptions are usually that the data are class-conditionally distributed multivariate normal and that the data used to train the classifier are representative of the area of interest. This second assumption actually has several parts. The assumption is made that in the process of training, all classes present in the scene are found, and all spectral subclasses of each class are also represented in the training data. Furthermore, the parameters of the distribution of each subclass are also assumed to be known from the training data. Each pixel is assumed to come from one of the training classes, and also is assumed to be entirely of one cover type.

In actual practice, these assumptions are not met. The number of spectral classes in the area is not known and clustering or some other method is used to determine the number of subclasses, in addition to estimating the statistics of those subclasses. Some of these methods also lead to non-normal subclasses. In particular, the clustering algorithm available through LARSYS truncates the tails of the subclass distributions and so leads to non-normal distributions.

There are also questions relating to a single picture element. A single pixel in Landsat data covers an area approximately 80 meters by 50 meters. More than one cover type may be present in this area and result in a "mixture pixel" observation. It is not clear how the distribution of the spectral response of mixture pixels can be related to the distribution of the spectral response of "pure pixels."

---

\*This work was sponsored by NASA Contract NAS9-15466.

There has been much speculation in the remote sensing community as to the effect of the non-satisfaction of the basic assumptions. Whenever new algorithms are brought forth, the old questions are raised again, indicating that there is insufficient understanding of the interaction of the real attributes of the data and the theory of the algorithms. At times it is not clear whether a particular result is due to aspects of the algorithm or to the extent the data set deviates from the assumptions.

In testing new algorithms, deviations from the assumptions may obscure the action of the new process. One way to clarify the situation is to apply the algorithm first to a data set satisfying the assumptions.

Such a data set could be obtained artificially, through simulation. The analyst could then know: how many classes exist in the data; the true distributions of the classes, including normality if desired; the observations could really be independent; and no pixel would be a "mixture pixel." New algorithms could be studied on such a data set with the knowledge that any "strange" effects are indeed algorithm rather than data problems.

In many cases where simulated data have been used in the past, the data were too artificial, in the sense that all aspects of the image were controlled, removing the natural variation in object size, position, and relationship which occur in real data. This limited the use of the simulated data sets in testing new algorithms.

The natural spatial information occurring in multispectral data could be retained in a simulated image by spatially basing the simulation on a classification. It would be even better to base the simulated data on a digitized "ground truth" map if the spectral characteristics of the cover types were known. By basing the simulation on a classification, the number of classes, their exact distributions, and the class of each pixel in the

area are known. If the classification was sufficiently accurate, then the spatial information held in the classification map will be close to the actual cover type map and actual spatial content of the original data. For each pixel in the area, a random vector distributed according to the pixel's class statistics could be generated. This becomes the simulated data vector.

### Statistical Background

From the classification chosen as a basis for the simulation, the following are known: the number of classes  $K$ , the set of classes  $\{\omega_i, i=1, \dots, K\}$ , the class distributions  $\{f(\omega_i), i=1, \dots, K\}$ , their means and covariances  $\{\mu_i$  and  $\Sigma_i, i=1, \dots, K\}$ , the number of channels  $p$ , and the class of every pixel in the scene.

From classificial statistics:

(1) Let  $X:p \times 1$ ,  $A:p \times p$  and  $b:p \times 1$ .

If  $X \sim N(0, I_p)$ , then  $Y = AX + b \sim N(b, AI_pA^T = AA^T)$

(where  $I_p$  is the identity matrix having dimensionality  $p$ ).

(2) Let  $\Sigma$  be a symmetric, positive definite matrix. Then there exists  $A$ , such that

$$AA^T = \Sigma \quad (A \text{ is denoted } \Sigma^{\frac{1}{2}})$$

To simulate a pixel which was a member of class  $i$  in the base classification,  $N(0, I_p)$  (the random vector for each pixel is independent of other vectors) is generated. (See Appendix I.) Next  $Y = \Sigma_i^{\frac{1}{2}} X + \mu_i$  is calculated; it is then a random vector from the population  $N(\mu_i, \Sigma_i)$ . This process is repeated for each pixel of the base classification and the random vectors thus generated are stored appropriately, i.e., so as to correspond to their simulated spatial location.

The program requires as an input a classification map stored on a results tape. The results tape has the class statistics

for p-dimensions also stored on it. The program, then, uses the results map and the stored statistics to generate a p-dimensional data set, which is stored on a user specified output tape in LARSYS format.

Appendix I provides a mathematical derivation related to the generation of normally distributed samples. Appendix II provides the Fortran program listing for the simulation program. Appendix III provides the C program listing for the same program.



5

APPENDIX I

# APPENDIX I

Let  $U_1$  and  $U_2$  be two random variables independent and identically distributed Uniform  $(0,1)$ .

$$\text{Then let } Z_1 = (-2 \ln U_1)^{\frac{1}{2}} \cos 2\pi U_2$$

$$\text{and } Z_2 = (-2 \ln U_1)^{\frac{1}{2}} \sin 2\pi U_2$$

then  $Z_1$  and  $Z_2$  are independent and identically distributed normal  $(0,1)$ .

Proof:

$$f(U_1, U_2) = \begin{cases} 1 & 0 < U_1 < 1, 0 < U_2 < 1 \\ 0 & \text{otherwise} \end{cases}$$

is the probability density function of two independent uniforms.

$$U_1 = \exp [-\frac{1}{2}(Z_1^2 + Z_2^2)]$$

$$U_2 = \frac{1}{2\pi} \arctan\left(\frac{Z_2}{Z_1}\right)$$

The Jacobian of the transformation is:

$$J = -\frac{1}{2\pi} \exp[-\frac{1}{2}(Z_1^2 + Z_2^2)]$$

$$f(Z_1, Z_2) = f(U_1, U_2) \cdot |J|$$

$$= \frac{1}{2\pi} \exp [-\frac{1}{2}(Z_1^2 + Z_2^2)] \quad 0 < [\exp -\frac{1}{2}(Z_1^2 + Z_2^2)] < 1$$

$$0 < \frac{1}{2\pi} \arctan\left(\frac{Z_2}{Z_1}\right) < 1$$

$$= 0 \quad \text{otherwise}$$

$$\therefore f(Z_1) \sim N(0,1) \quad f(Z_2) \sim N(0,1)$$

The side conditions give  $-\infty < Z_1 < \infty$ ,  $-\infty < Z_2 < \infty$ . Strictly speaking,  $Z_1$  cannot equal zero; however,  $\text{prob}(Z_1 = 0) = 0$  as we are working with continuous densities.

To test the effectiveness of the pseudo random vectors in the multivariate case, random vectors distributed  $N(0, I_p)$  were generated and then tested with a Kolmogorov-Smirnov test. Since the multivariate normal cdf is difficult to evaluate, the sum of squares was calculated and compared to the  $\chi_p^2$  distribution.

For sample sizes greater than 100, the pseudo random vectors were distributed properly. For sample sizes less than 100, the K-S test is not valid. Since we would generally (over an entire area) be working with more than 100 points per class, this was not pursued further.

In addition, the sample covariance matrices were tested for homogeneity against the true class statistics. For sample runs of up to 2000 points, there were not significant differences at the  $\alpha = 0.10$  level.

## APPENDIX II

FILE: SWRITE FORTRAN A PUROUE / LARS 3031

[illegible]

FILE: SWRITE FORTRAN 4 PURDUE / LARS 3031

C 150 CONTINUE

```

CALL TOPWR:12,800,IER,IOREC1
IF:IER.NE.0 WRITE:16,234:IER
IF:IER.GT.0 GO TO 31C
DO 50 MA=1,NOCLAS
CLAPNT:MA=0
DO 50 MB=1,NOCHAN
IMEAN:MA,MB=0
RMEAN:MA,MB=0.0
DO 50 MC=1,NOCHAN
IVAR:MA,MB,MC=0
50 RVAR:MA,MB,MC=C.0
LNWRT = 0
55 READ:11,J,K,LINENO,:PNTCLS:IX,IX=1,NCPNTS1
IF:J.GT.6 GO TO 55
LNWRT=LNWRT+1
IF:MOD:LNWRT,25.EQ.C WRITE:16,57:LNWRT,NOLINE
57 FORMAT:5X,I4,' LINES OUT OF 1,14, ARE COMPLETED')

```

\*\*\*\*\*  
GENERATE AND WRITE DATA POINTS  
\*\*\*\*\*

```

60 I2=ILIN:21
DATOUT:1=L1:11
DATOUT:2=L1:21
I2=32767
DATOUT:3=L1:11
DATOUT:4=L1:21
I2=5
ICOUNT=4
DO 90 IX=1,NCPNTS
ICOUNT=ICOUNT+1
I2=PNTCLS:IX
L1:11=.FALSE.
IPOL=:I2-1:1:NUCHAN
IBEG=:I2-1:1:NDCOMP
K=IBEG
DO 65 IY=1,NOCHAN
DO 65 IZ=1,IY
K=K+1
R:IY,I2=Z:K
IF:IY.EQ.IZ GO TO 65
B:IZ,IY=0.0
65 CONTINUE
DO 70 IY=1,NOCH
CALL RANDU:1:START:IY,NXINP,A2:IY11
1:START:IY=NXINP
CALL RANDU:1:START:IY,NXINP,A:IY11
1:START:IY=NXINP
A:IY11=5:RT:-2.*ALOG:42:IY11*CUS:c.28312*A:IY11
70 CONTINUE
CLAPNT:I2=CLAPNT:I2+1
DO 80 IY=1,NOCHAN
DATA:IY=C.0
IQ=NOPOOL*NDCOMP*IPOL*IY
DO 75 IZ=1,NOCHAN
DATA:IY=DATA:IY+B:IY,I2+A:I21
DATA:IY=DATA:IY+Z:101
INTDAT=DATA:IY+.5
IF:INTDAT.LT.C1 INTDAT=G
IF:INTDAT.GT.255 INTDAT=255
1:STAT:IY=INTDAT
DATOUT:IY-1:1:NDCHAN*ICOUNT+LOGDAT:21
DO 92 IZ=1,6
DATOUT:IY-1:1:NDCHAN*ICOUNT+IZ1=.FALSE.
92 CONTINUE
DO 90 II=1,NOCHAN
IMEAN:I2,II=IMEAN:I2,II+1:STAT:II1
DO 90 JJ=1,NOCHAN
IVAR:I2,II,JJ=IVAR:I2,II,JJ+1:STAT:II1+1:STAT:JJ1
90 CONTINUE
NOBYTE=4*NOCHAN*NDCHAN
CALL TOPWR:12,NOBYTE,IER,DATOUT:1
IF:IER.NE.0 WRITE:16,234:IER
IF:IER.GT.0 GO TO 31C
GO TO 55
95 CONTINUE

```

SWR C2380  
SWR C2390  
SWR C2400  
SWR C2410  
SWR C2420  
SWR C2430  
SWR C2440  
SWR C2450  
SWR C2460  
SWR C2470  
SWR C2480  
SWR C2490  
SWR C2500  
SWR C2510  
SWR C2520  
SWR C2530  
SWR C2540  
SWR C2550  
SWR C2560  
SWR C2570  
SWR C2580  
SWR C2590  
SWR C2600  
SWR C2610  
SWR C2620  
SWR C2630  
SWR C2640  
SWR C2650  
SWR C2660  
SWR C2670  
SWR C2680  
SWR C2690  
SWR C2700  
SWR C2710  
SWR C2720  
SWR C2730  
SWR C2740  
SWR C2750  
SWR C2760  
SWR C2770  
SWR C2780  
SWR C2790  
SWR C2800  
SWR C2810  
SWR C2820  
SWR C2830  
SWR C2840  
SWR C2850  
SWR C2860  
SWR C2870  
SWR C2880  
SWR C2890  
SWR C2900  
SWR C2910  
SWR C2920  
SWR C2930  
SWR C2940  
SWR C2950  
SWR C2960  
SWR C2970  
SWR C2980  
SWR C2990  
SWR C3000  
SWR C3010  
SWR C3020  
SWR C3030  
SWR C3040  
SWR C3050  
SWR C3060  
SWR C3070  
SWR C3080  
SWR C3090  
SWR C3100  
SWR C3110  
SWR C3120  
SWR C3130  
SWR C3140  
SWR C3150  
SWR C3160

FILE: SWRITE FORTRAN 4 PURDUE / LARS 3031

```

      IBEG=1
      C*****
      C FACTOR COVARIANCE MATRICES
      C*****
      DO 30 IX=1,NOPOOL
      IDONE=IBEG+NOCOMP-1
      K=0
      DO 20 IY=IBEG,IDONE
      K=K+1
      20 A(K)=Z(IY)
      CALL MFSO(4,NOCHAN,EPS,IER)
      IF IER.EQ.-1 GO TO 300
      IF IER.GE.1 GO TO 310
      K=0
      DO 25 IY=IBEG,IDONE
      K=K+1
      25 Z(IY)=A(K)
      30 IBEG=IBEG+NOCOMP
      C*****
      C GENERATE STARTING POINTS
      C*****
      29 WRITE(10,31)
      31 FORMAT(5X,'DO YOU WANT TO SPECIFY THE STARTING POINTS FOR THE'/5X
      $,'RANDOM NUMBER GENERATOR? (TYPE YES OR NO)')
      READ(10,32)INPUT
      32 FORMAT(4I)
      IF INPUT.EQ.NO GO TO 36
      IF INPUT.EQ.YES GO TO 33
      GO TO 29
      33 DO 39 IX=1,NOCHAN
      WRITE(10,41)IX
      41 FORMAT(5X,'SPECIFY STARTING POINT FOR CHANNEL',I3/5X,'(TYPE A NINE
      $ DIGIT ODD NUMBER)')
      READ(10,42)ISTART:IX)
      42 FORMAT(19)
      39 CONTINUE
      GO TO 43
      36 CALL GTSERL:ISERL)
      ISERL=:ISERL/10)+0.1
      DO 40 I=1,NOCH
      ISERL=ISERL+ICCCCC
      ISTART:I=ISERL
      40 CONTINUE
      43 WRITE(6,34)
      34 FORMAT(77/5X,'STARTING POINTS FOR RANDOM NUMBER GENERATOR'//)
      DO 44 I=1,NOCHAN
      WRITE(6,35)I,ISTART:I)
      35 FORMAT(5X,'STARTING POINT FOR CHANNEL ',I2,' IS ',I9)
      44 CONTINUE
      C*****
      C READ CLASSIFICATIONS
      C*****
      IDREC:1)=TAPEND
      IDREC:2)=JFILE
      IDREC:3)=RUNNO
      NOLD = IDREC:5)
      IDREC:5) = NOCHAN
      IDREC:6) = 4*NOPTS + 51/4)
      NOSAM = IDREC:6)
      IDREC:7) = FLGT
      DO 141 I=1,3
      IDREC:11+16) = DATE:1)
      141 CONTINUE
      IDREC:20) = NOLINE
      DO 145 I=1,NOCHAN
      INEW = PETVC3:I)
      DO 145 I12 = 1,5
      FRQCAL:I12,I1) = FRQCAL:I12,INEW)
      145 CONTINUE
      LIP = NOCHAN + 1
      DO 150 I1 = LIP,NOLD
      DO 150 I12 = 1,5
      FRQCAL:I12,I1) = 0.0

```

SARC01590  
 SARC01600  
 SARC01610  
 SARC01620  
 SARC01630  
 SARC01640  
 SARC01650  
 SARC01660  
 SARC01670  
 SARC01680  
 SARC01690  
 SARC01700  
 SARC01710  
 SARC01720  
 SARC01730  
 SARC01740  
 SARC01750  
 SARC01760  
 SARC01770  
 SARC01780  
 SARC01790  
 SARC01800  
 SARC01810  
 SARC01820  
 SARC01830  
 SARC01840  
 SARC01850  
 SARC01860  
 SARC01870  
 SARC01880  
 SARC01890  
 SARC01900  
 SARC01910  
 SARC01920  
 SARC01930  
 SARC01940  
 SARC01950  
 SARC01960  
 SARC01970  
 SARC01980  
 SARC01990  
 SARC02000  
 SARC02010  
 SARC02020  
 SARC02030  
 SARC02040  
 SARC02050  
 SARC02060  
 SARC02070  
 SARC02080  
 SARC02090  
 SARC02100  
 SARC02110  
 SARC02120  
 SARC02130  
 SARC02140  
 SARC02150  
 SARC02160  
 SARC02170  
 SARC02180  
 SARC02190  
 SARC02200  
 SARC02210  
 SARC02220  
 SARC02230  
 SARC02240  
 SARC02250  
 SARC02260  
 SARC02270  
 SARC02280  
 SARC02290  
 SARC02300  
 SARC02310  
 SARC02320  
 SARC02330  
 SARC02340  
 SARC02350  
 SARC02360  
 SARC02370

5  
4  
3  
2  
1

A 4x5 grid of 20 small, stylized icons. The icons include: a car, a house, a tree, a flower, a leaf, a fruit, a vegetable, a person, a group of people, a building, a bridge, a road, a river, a mountain, a cloud, a sun, a moon, a star, a heart, a cross, and a circle.

[illegible]

555  
555  
555  
555  
555  
555  
555  
555

[illegible]

00960000000000000000

5  
4  
3  
2  
1

555  
555  
555  
555  
555  
555  
555  
555

5  
4  
3  
2  
1

五  
三  
二  
一

00708101 UCIR 2755

人々々々  
 世々々々  
 万々々々  
 千々々々  
 百々々々  
 十々々々  
 一々々々

55  
54  
53  
52  
51  
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24  
23  
22  
21  
20  
19  
18  
17  
16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

五、六、七、八、九、十、十一、十二、十三、十四、十五、十六、十七、十八、十九、二十、二十一、二十二、二十三、二十四、二十五、二十六、二十七、二十八、二十九、三十、三十一、三十二、三十三、三十四、三十五、三十六、三十七、三十八、三十九、四十、四十一、四十二、四十三、四十四、四十五、四十六、四十七、四十八、四十九、五十、五十一、五十二、五十三、五十四、五十五、五十六、五十七、五十八、五十九、六十、六十一、六十二、六十三、六十四、六十五、六十六、六十七、六十八、六十九、七十、七十一、七十二、七十三、七十四、七十五、七十六、七十七、七十八、七十九、八十、八十一、八十二、八十三、八十四、八十五、八十六、八十七、八十八、八十九、九十、九十一、九十二、九十三、九十四、九十五、九十六、九十七、九十八、九十九、一百。

SAR C 1170  
SAR C 1180

554  
553  
552  
551  
550  
549  
548  
547  
546  
545  
544  
543  
542  
541  
540  
539  
538  
537  
536  
535  
534  
533  
532  
531  
530  
529  
528  
527  
526  
525  
524  
523  
522  
521  
520  
519  
518  
517  
516  
515  
514  
513  
512  
511  
510  
509  
508  
507  
506  
505  
504  
503  
502  
501  
500  
499  
498  
497  
496  
495  
494  
493  
492  
491  
490  
489  
488  
487  
486  
485  
484  
483  
482  
481  
480  
479  
478  
477  
476  
475  
474  
473  
472  
471  
470  
469  
468  
467  
466  
465  
464  
463  
462  
461  
460  
459  
458  
457  
456  
455  
454  
453  
452  
451  
450  
449  
448  
447  
446  
445  
444  
443  
442  
441  
440  
439  
438  
437  
436  
435  
434  
433  
432  
431  
430  
429  
428  
427  
426  
425  
424  
423  
422  
421  
420  
419  
418  
417  
416  
415  
414  
413  
412  
411  
410  
409  
408  
407  
406  
405  
404  
403  
402  
401  
400  
399  
398  
397  
396  
395  
394  
393  
392  
391  
390  
389  
388  
387  
386  
385  
384  
383  
382  
381  
380  
379  
378  
377  
376  
375  
374  
373  
372  
371  
370  
369  
368  
367  
366  
365  
364  
363  
362  
361  
360  
359  
358  
357  
356  
355  
354  
353  
352  
351  
350  
349  
348  
347  
346  
345  
344  
343  
342  
341  
340  
339  
338  
337  
336  
335  
334  
333  
332  
331  
330  
329  
328  
327  
326  
325  
324  
323  
322  
321  
320  
319  
318  
317  
316  
315  
314  
313  
312  
311  
310  
309  
308  
307  
306  
305  
304  
303  
302  
301  
300  
299  
298  
297  
296  
295  
294  
293  
292  
291  
290  
289  
288  
287  
286  
285  
284  
283  
282  
281  
280  
279  
278  
277  
276  
275  
274  
273  
272  
271  
270  
269  
268  
267  
266  
265  
264  
263  
262  
261  
260  
259  
258  
257  
256  
255  
254  
253  
252  
251  
250  
249  
248  
247  
246  
245  
244  
243  
242  
241  
240  
239  
238  
237  
236  
235  
234  
233  
232  
231  
230  
229  
228  
227  
226  
225  
224  
223  
222  
221  
220  
219  
218  
217  
216  
215  
214  
213  
212  
211  
210  
209  
208  
207  
206  
205  
204  
203  
202  
201  
200  
199  
198  
197  
196  
195  
194  
193  
192  
191  
190  
189  
188  
187  
186  
185  
184  
183  
182  
181  
180  
179  
178  
177  
176  
175  
174  
173  
172  
171  
170  
169  
168  
167  
166  
165  
164  
163  
162  
161  
160  
159  
158  
157  
156  
155  
154  
153  
152  
151  
150  
149  
148  
147  
146  
145  
144  
143  
142  
141  
140  
139  
138  
137  
136  
135  
134  
133  
132  
131  
130  
129  
128  
127  
126  
125  
124  
123  
122  
121  
120  
119  
118  
117  
116  
115  
114  
113  
112  
111  
110  
109  
108  
107  
106  
105  
104  
103  
102  
101  
100  
99  
98  
97  
96  
95  
94  
93  
92  
91  
90  
89  
88  
87  
86  
85  
84  
83  
82  
81  
80  
79  
78  
77  
76  
75  
74  
73  
72  
71  
70  
69  
68  
67  
66  
65  
64  
63  
62  
61  
60  
59  
58  
57  
56  
55  
54  
53  
52  
51  
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24  
23  
22  
21  
20  
19  
18  
17  
16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

55  
54  
53  
52  
51  
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24  
23  
22  
21  
20  
19  
18  
17  
16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

55  
54  
53  
52  
51  
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24  
23  
22  
21  
20  
19  
18  
17  
16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
13

555  
122  
722  
777  
111  
222  
999  
777

0  
1  
2  
3  
3  
3  
1  
1  
7

55  
 56  
 57  
 58  
 59  
 60  
 61  
 62  
 63  
 64  
 65  
 66  
 67  
 68  
 69  
 70  
 71  
 72  
 73  
 74  
 75  
 76  
 77  
 78  
 79  
 80  
 81  
 82  
 83  
 84  
 85  
 86  
 87  
 88  
 89  
 90  
 91  
 92  
 93  
 94  
 95  
 96  
 97  
 98  
 99  
 100  
 101  
 102  
 103  
 104  
 105  
 106  
 107  
 108  
 109  
 110  
 111  
 112  
 113  
 114  
 115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159  
 160  
 161  
 162  
 163  
 164  
 165  
 166  
 167  
 168  
 169  
 170  
 171  
 172  
 173  
 174  
 175  
 176  
 177  
 178  
 179  
 180  
 181  
 182  
 183  
 184  
 185  
 186  
 187  
 188  
 189  
 190  
 191  
 192  
 193  
 194  
 195  
 196  
 197  
 198  
 199  
 200  
 201  
 202  
 203  
 204  
 205  
 206  
 207  
 208  
 209  
 210  
 211  
 212  
 213  
 214  
 215  
 216  
 217  
 218  
 219  
 220  
 221  
 222  
 223  
 224  
 225  
 226  
 227  
 228  
 229  
 230  
 231  
 232  
 233  
 234  
 235  
 236  
 237  
 238  
 239  
 240  
 241  
 242  
 243  
 244  
 245  
 246  
 247  
 248  
 249  
 250  
 251  
 252  
 253  
 254  
 255  
 256  
 257  
 258  
 259  
 260  
 261  
 262  
 263  
 264  
 265  
 266  
 267  
 268  
 269  
 270  
 271  
 272  
 273  
 274  
 275  
 276  
 277  
 278  
 279  
 280  
 281  
 282  
 283  
 284  
 285  
 286  
 287  
 288  
 289  
 290  
 291  
 292  
 293  
 294  
 295  
 296  
 297  
 298  
 299  
 300  
 301  
 302  
 303  
 304  
 305  
 306  
 307  
 308  
 309  
 310  
 311  
 312  
 313  
 314  
 315  
 316  
 317  
 318  
 319  
 320  
 321  
 322  
 323  
 324  
 325  
 326  
 327  
 328  
 329  
 330  
 331  
 332  
 333  
 334  
 335  
 336  
 337  
 338  
 339  
 340  
 341  
 342  
 343  
 344  
 345  
 346  
 347  
 348  
 349  
 350  
 351  
 352  
 353  
 354  
 355  
 356  
 357  
 358  
 359  
 360  
 361  
 362  
 363  
 364  
 365  
 366  
 367  
 368  
 369  
 370  
 371  
 372  
 373  
 374  
 375  
 376  
 377  
 378  
 379  
 380  
 381  
 382  
 383  
 384  
 385  
 386  
 387  
 388  
 389  
 390  
 391  
 392  
 393  
 394  
 395  
 396  
 397  
 398  
 399  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 413  
 414  
 415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 426  
 427  
 428  
 429  
 430  
 431  
 432  
 433  
 434  
 435  
 436  
 437  
 438  
 439  
 440  
 441  
 442  
 443  
 444  
 445  
 446  
 447  
 448  
 449  
 450  
 451  
 452  
 453  
 454  
 455  
 456  
 457  
 458  
 459  
 460  
 461  
 462  
 463  
 464  
 465  
 466  
 467  
 468  
 469  
 470  
 471  
 472  
 473  
 474  
 475  
 476  
 477  
 478  
 479  
 480  
 481  
 482  
 483  
 484  
 485  
 486  
 487  
 488  
 489  
 490  
 491  
 492  
 493  
 494  
 495  
 496  
 497  
 498  
 499  
 500  
 501  
 502  
 503  
 504  
 505  
 506  
 507  
 508  
 509  
 510  
 511  
 512  
 513  
 514  
 515  
 516  
 517  
 518  
 519  
 520  
 521  
 522  
 523  
 524  
 525  
 526  
 527  
 528  
 529  
 530  
 531  
 532  
 533  
 534  
 535  
 536  
 537  
 538  
 539  
 540  
 541  
 542  
 543  
 544  
 545  
 546  
 547  
 548  
 549  
 550  
 551  
 552  
 553  
 554  
 555  
 556  
 557  
 558  
 559  
 560  
 561  
 562  
 563  
 564  
 565  
 566  
 567  
 568  
 569  
 570  
 571

555  
222  
777  
777  
111  
333  
888  
777

55  
54  
53  
52  
51  
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24  
23  
22  
21  
20  
19  
18  
17  
16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

U  
U  
U  
U  
U  
U  
U  
U  
U

155  
22  
20  
10  
14  
45  
55

55  
54  
53  
52  
51  
50  
49  
48  
47  
46  
45  
44  
43  
42  
41  
40  
39  
38  
37  
36  
35  
34  
33  
32  
31  
30  
29  
28  
27  
26  
25  
24  
23  
22  
21  
20  
19  
18  
17  
16  
15  
14  
13  
12  
11  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

SERIALS

0001551100

[illegible]

SAR C1570

3MKU136C



FILE: SWRITE FORTRAN A PURDUE / LARS 3C31

\*\*\*\*\*  
 WRITTEN BY: BILL PFAPP  
 EDITED BY: MARWAN MUASHER JUNE 14, 1980  
 \*\*\*\*\*

\*\*\*\*\*  
 THIS PROGRAM GENERATES SIMULATED DATA BASED ON A  
 CLASSIFICATION MAP OR A GROUND TRUTH MAP. EACH PIXEL  
 GENERATED THUS COMES FROM A KNOWN CLASS DISTRIBUTION. THE  
 METHOD USED IS AS FOLLOWS:  
 1. A GOOD CLASSIFICATION IS CHOSEN AS A BASE FOR  
 SIMULATED DATA.  
 2. FROM THIS CLASSIFICATION WE KNOW THE NUMBER OF CLASSES, THE  
 CLASS STATISTICS, AND THE CLASS OF EACH PIXEL IN THE  
 AREA CLASSIFIED.  
 3. A STREAM OF UNIFORM RANDOM NUMBERS IS GENERATED FOR  
 EACH CHANNEL. THEY ARE CHANGED TO NORMAL (0,1) DEVIATES.  
 4. FOR EACH PIXEL, A RANDOM N(0,1) VECTOR IS TRANSFORMED TO  
 BE DISTRIBUTED ACCORDING TO THE CLASS STATISTICS OF THAT  
 PIXEL. THIS IS THE SIMULATED DATA VECTOR.  
 5. AS EACH LINE IS COMPLETED, IT IS WRITTEN TO AN OUTPUT TAPE.  
 TO RUN THE PROGRAM, YOU NEED TO HAVE THE FOLLOWING  
 EXEC FILE ON YOUR DISK:

```
GETDISK LARSYS
GETDISK DVSYS
GLOBAL TXTLIB CMSLIB FORTRAN SSP370
FILEDEF 6 PRINTER
FILEDEF 16 TERMINAL
FILEDEF 12 TAP2
FILEDEF 11 TAP1:RECFM VS LRECL 1500 BLKSIZE 1500
LOAD SWRITE GLOCOP MMTAPE TAP0P 300VAL GTSERL GTDATE MFSD
RANDU WRTMTX
START SWRITE
```

THE PROGRAM WILL ASK FOR INFORMATION SUCH AS  
 TAPE NUMBERS, FILE NUMBERS, .ETC. FROM HERE ON, IT  
 SHOULD BE EASY TO FOLLOW.

\*\*\*\*\*  
 VARIABLES USED IN TPRINT

```
A = COVARIANCE STORAGE FOR FACTORING
AREANO = AREA NUMBER OF CLASSIFICATION
B = COVARIANCE STORAGE FOR MULTIPLICATION
DATA = DATA POINT STORAGE
DATAVAL = LINE NUMBER AND ROLL PARAMETER
ICAL = CALIBRATION INFORMATION
IDREC = IDENTIFICATION RECORD STORAGE
ISTART = STARTING POINTS FOR GAUSS
LOGDAT = DATA POINTS IN LOGICAL FORMAT
NOCHAN = NUMBER OF CHANNELS IN CLASSIFICATION
NOCLAS = NUMBER OF CLASSES IN ORIGINAL STATISTICS
NOFLDS = NUMBER OF TEST FIELDS
NOPOL = NUMBER OF POOLED CLASSES
PNTCLS = CLASSIFICATIONS ARRAY
Z = STATISTICS STORAGE
```

\*\*\*\*\*  
 INITIALIZATION

```
INTEGER*2 I2,INTDAT,ICAL(3),ILIN(2),PNTCLS(1000),ISTAT(4),
  $ FEIVC(300)
LOGICAL*1 L1(2),LOGDAT(2),LCAL(6),DATELT(12000)
REAL*4 A(78),A2(12),Z(2700),R(12),DATA(12),
  $ RMEAN(30),I2VAR(30),I2Z(2700),FMICAL(5,30)
INTEGER*4 ISTART(12),EDS,INFO(17),AREANO,IDREC(200),TAPEND,THREE,
  $ CLAPNT(30),I2EAN(30),I2VAR(30),I2YES,NO,DATE(31)
INTEGER*4 RUNNC,FLGT
EQUIVALENCE (I2,L1),INTDAT,LOGDAT,ICAL,LCAL,ILINRT,ILIN
EQUIVALENCE (FROCAL(1,1),IDREC(51))
DATA EDS,S,AM /EDS,1,C,C,C /
DATA YES,NO,THREE /YES,1,NO,1,3*/
```

SWR00010  
 SWR00020  
 SWR00030  
 SWR00040  
 SWR00050  
 SWR00060  
 SWR00070  
 SWR00080  
 SWR00090  
 SWR00100  
 SWR00110  
 SWR00120  
 SWR00130  
 SWR00140  
 SWR00150  
 SWR00160  
 SWR00170  
 SWR00180  
 SWR00190  
 SWR00200  
 SWR00210  
 SWR00220  
 SWR00230  
 SWR00240  
 SWR00250  
 SWR00260  
 SWR00270  
 SWR00280  
 SWR00290  
 SWR00300  
 SWR00310  
 SWR00320  
 SWR00330  
 SWR00340  
 SWR00350  
 SWR00360  
 SWR00370  
 SWR00380  
 SWR00390  
 SWR00400  
 SWR00410  
 SWR00420  
 SWR00430  
 SWR00440  
 SWR00450  
 SWR00460  
 SWR00470  
 SWR00480  
 SWR00490  
 SWR00500  
 SWR00510  
 SWR00520  
 SWR00530  
 SWR00540  
 SWR00550  
 SWR00560  
 SWR00570  
 SWR00580  
 SWR00590  
 SWR00600  
 SWR00610  
 SWR00620  
 SWR00630  
 SWR00640  
 SWR00650  
 SWR00660  
 SWR00670  
 SWR00680  
 SWR00690  
 SWR00700  
 SWR00710  
 SWR00720  
 SWR00730  
 SWR00740  
 SWR00750  
 SWR00760  
 SWR00770  
 SWR00780  
 SWR00790

## APPENDIX III

```

/*****
*
* Read data from LARS Results Tape
* and simulate data from it
* using the Box Muller relationship
*
*****/

* Swrite.c has been translated from
* the Lars Fortran Version of Swrite into
* the language 'c' for the Unix O. S.
* run on the DEC PDP-11/45
*
*****/

* Variables used in Swrite:
*
*   a      == Covariance storage for factorings
*   b      == Covariance storage for multiplication
*   data   == Data point storage
*   nochan == Number of channels in Classification
*   noclas == Number of classes in original statistics
*   norool == Number of rooted classes
*   nrtcls == Classifications array
*   z      == Statistics storage
*
*****/

* compile with      cc swrite.c -lp -lq /usr/lib/pdslib
*
*****/

* Initialize all variables used in swrite
* External variables are available to all functions
* within which they are declared
*/
main() {
    extern int noclas, nochan, norool, fd1, nrtcls, noline, nrcnum;
    extern int ier, fd2;
    extern float eps, upper[5], lower[5];
    int i, k, fetvc[5], debus, info[17], nrtcls[1000], j, ipol;
    int ii, idone, ix, ma, mb, mc;
    int noch, nocomp, ictor, iend, ix, ibes;
    int lnumt, i2, icount, iz, iq, intdat, istat[6], ie, io, it, ninc, no, in;
    int nosam, noc, rfd1;
    int fd3, fd4, err5;
    char buf[1500], *delim, *c1, *c2, *c3, datout[2100], obuf[2100];
    char rname[30], t1[41];
    char mufile[30], cmfile[30], orname[30];
    float z[610], nrtnt, r;
    float b[6][6];
    double rmean[31][6];
    float data[13], z2[610];
    float claent[35];
    float mobuf[30];
    double ivar[31][6][6], revar, imean[31][6], t1float, t2float, remean, semean;
    double temp1, temp2, temp0, a[20], a2[6];
    double sqrt(), log(), cos(), fmod(), s, t, f;

/* begin main program
* if debug is set to minus one the following is

```

```
* printed out for verification on usr terminal:
*   nochan, nopool, fetuc3, info, entcls, noents, noline, z, upper, lower;
*/
debus = 1;
eps = .00001;
/* read records 2,4,5,&6 off results tape*/
/* skip records 1,3 */
readito5(buf, fetuc3, z, info);
noch = (((nochan+1)/2)*2);
nocomp = (nochan*(nochan+1)/2);
istop = nocomp * nopool;
iend = nochan * nopool + istop;
nosam = 4 * ((noents + 9)/4);
if(nochan>5) {
    printf("Number of channels is %d but internal", nochan);
    printf(" storage only allows 5\n");
    printf("Execution terminated abnormally \n");
    exit();
}
if(nopool>29) {
    printf("Number of pooled classes is %d but internal", nopool);
    printf(" storage only allows 29\n");
    printf("Execution terminated abnormally \n");
    exit();
}
if(noents>=1000) {
    printf("Number of points per line is %d but internal", noents);
    printf(" storage only allows 1000\n");
    printf("Execution terminated abnormally \n");
    exit();
}
printf("nochan=%d, nopool=%d\n", nochan, nopool);
if(debus == -1) {
    for(k=0; k<nochan; ++k)
        printf("fetuc3[%d] = %d \n", k, fetuc3[k]);
    for(k=0; k<iend; k = k+10) {
        for(j=k; j<=k+9; ++j)
            printf(" %f", z[j]);
        printf("\n");
    }
    printf("\nField size:");
    printf("\n      line %d to %d with interval %d",
        info[4], info[5], info[6]);
    printf("\n      cols %d to %d with interval %d\n",
        info[7], info[8], info[9]);
    printf("Number of lines classified is %d\n", noline);
    printf("Number of points classified per line is %d\n", noents);
    for(j=0; j<nochan; ++j) {
        printf("upper[%d] = %f ", j, upper[j]);
        printf(" lower[%d] = %f \n", j, lower[j]);
    }
}
read6rec(buf, entcls);
if(debus == -1) {
    printf("\nFirst line of record %d follows\n", recnum);
    for(j=0; j<noents; j=j+10) {
        for(k=j; k<=j+9; ++k)
            printf(" %d", entcls[k]);
        printf("\n");
    }
}
```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

)
/* make a copy of statistics array z */
for(ix=1; ix<=iend; ++ix)
    z2[ix] = z[ix-1];
/* Create output file */
sets("Specify linerrinter output file pathname: ", orname);
fd2 = creat(orname, 0777);
if(fd2<0)
    printf("Cannot creat lr output file %s\n", orname);
rfd1 = 1;
sets("Specify simulated data PDS file pathname: ", rname);
sets("enter a 40 char title:\n", ttl);
sets("Specify mean vector file pathname: ", mvfile);
sets("Specify covariance matrix file pathname: ", cmfile);
rdsorec(rfd1, rname);
outfmt(rfd1, 1, nosam, noline, 8, nochan);
putttl(rfd1, ttl);
fd3 = creat(mvfile, 0777);
if(fd3<0)
    printf("Cannot creat mvfile %s (MAIN)\n", mvfile);
fd4 = creat(cmfile, 0777);
if(fd4<0)
    printf("Cannot creat cmfile %s (MAIN)\n", cmfile);
/* write output page one */
delim = "++++++";
cl = "Data Simulation Uses Box-Muller Relationship";
printf(fd2, "\n\n\n\n%s\n\n\n", delim, cl, delim);
printf(fd2, "\n Line %d to line %d with interval %d \n",
    info[4], info[5], info[6]);
printf(fd2, "\n Column %d to column %d with interval %d \n",
    info[7], info[8], info[9]);
printf(fd2, "\n channels used \n");
for(ix=1; ix<=nochan; ++ix)
    printf(fd2, " %d %f = %f\n", +tvec3[ix-1], lower[ix-1],
        upper[ix-1]);
/* output new page character */
printf(fd2, "\014");
/* Factor covariance matrices */
ibes = 1;
for(ix=1; ix<=nocomp; ++ix) {
    idone = ibes + nocomp - 1;
    k = 0;
    for(iy=ibes; iy<=idone; ++iy) {
        k = k + 1;
        a[k] = z[iy-1];
    }
/* call function to perform factorins */
mfsd(a);
if(ier == -1)
    printf(fd2, "Error -1\n");
if(ier>=1)
    printf(fd2, "Error st 1\n");
if((ier == -1) || (ier >= 1))
    exit(1);
k = 0;
for(iy=ibes; iy<=idone; ++iy) {
    k = k + 1;
    z[iy-1] = a[k];
}
ibes = ibes + nocomp;
}

```

```

    }
/* initialize random number generator */
srand(1);
/* initialize arrays */
for(ma=1; ma<=noclas; ++ma) {
    clasnt[ma] = 0.0;
    for(mb=1; mb<=nochan; ++mb) {
        imean[ma][mb] = 0.0;
        rmean[ma][mb] = 0.0;
        for(mc=1; mc<=nocomp; ++mc) {
            ivar[ma][mb][mc] = 0.0;
        }
    }
}

t = 25.0;
lnwrt = 0;
/* this while loop is repeated for each line in the classification */
while(recnum == 6) {
    lnwrt = lnwrt + 1;
    s = lnwrt;
    f = fmod(s, t);
    if(f == 0.0)
        printf("%d Lines out of %d are completed\n",
               lnwrt, noline);
    i2 = 0;
    icount = 4;
    for(ix=1; ix<=nosam; ++ix) {
        icount = icount + 1;
        i2 = entels[ix-1];
        ipol = (i2-1) * nochan;
        ibes = (i2-1) * nocomp;
        k = ibes;
        for(iy=1; iy<=nochan; ++iy) {
            for(iz=1; iz<=nocomp; ++iz) {
                k = k + 1;
                b[iy][iz] = z[k-1];
                if(iy != iz)
                    b[iz][iy] = 0.0;
            }
        }
        for(iy=1; iy<=nochan; ++iy) {
            a2[iy] = rand();
            a[iy] = rand();
            a2[iy] = a2[iy] / 32767.;
            a[iy] = a[iy] / 32767.;
            a[iy] = sqrt(-2.0 * log(a2[iy])) * cos(6.28318 * a[iy]);
        }
        clasnt[i2] = clasnt[i2] + 1.0;
        for(iy=1; iy<=nochan; ++iy) {
            data[iy] = 0.0;
            is = ipol * nocomp + ipol + iy;
            for(iz=1; iz<=nocomp; ++iz)
                data[iy] = data[iy] + b[iy][iz] * a[iz];
            data[iy] = data[iy] + z[i2-1];
            intdat = data[iy] + .5;
            if(intdat<0) intdat = 0;
            if(intdat>255) intdat = 255;
            istat[iy] = intdat;
            pos = (iy-1)*nosam + icount - 5;
            if(pos>2100) printf("datout internal buffer full\n");
        }
    }
}

```

```

        datout[pos] = intdat & 0377;
        for(i2=1; i2<=6; ++i2)
            datout[pos+i2] = 0;
    }
    for(ii=1; ii<=nochan; ++ii) {
        imean[i2][ii] = imean[i2][ii] + istat[ii];
        for(jj=1; jj<=nochan; ++jj) {
            temp0 = istat[ii];
            temp1 = istat[jj];
            ivar[i2][ii][jj] = ivar[i2][ii][jj] + temp0 * temp1;
        }
    }
    ii = 0;
    for(iy=0; iy<nosam; ++iy) {
        for(i2=0; i2<nochan; ++i2)
            obuf[ii++] = datout[nosam*i2 + iy];
    }
    pds1pos(ffd1, lnwrt-1);
    if(putline(ffdl, lnwrt-1, obuf, nosam*nochan) != 0)
        printf("Putline to PDS output failed\n");
    read6rec(buf, entcls);
}
/* end of while loop */
/*
for(i2=1; i2<=18; ++i2) {
    printf(fd2, "\n %d \n", i2);
    for(ii=1; ii<=nochan; ++ii) {
        for(jj=1; jj<=nochan; ++jj) {
            printf(fd2, " %f ", ivar[i2][ii][jj]);
        }
        printf(fd2, "\n");
    }
}
for(i2=1; i2<=18; ++i2) {
    printf(fd2, "\n");
    for(ii=1; ii<=nochan; ++ii) {
        printf(fd2, " %f \n", imean[i2][ii]);
    }
}
*/
for(ip=1; ip<=noclas; ++ip) {
    for(io=1; io<=nochan; ++io) {
        if(clarnt[ip]>0.0) {
            t1float = imean[ip][io];
            t2float = clarnt[ip];
            rmean[ip][io] = t1float/t2float;
        }
        for(it=1; it<=nochan; ++it) {
            if(clarnt[ip]>1.0) {
                repnt = clarnt[ip];
                revar = ivar[ip][io][it];
                remean = imean[ip][io];
                semean = imean[ip][it];
                temp0 = revar/(repnt-1.0);
                temp1 = remean/repnt;
                temp2 = semean/(repnt-1.0);
                ivar[ip][io][it] = temp0 - (temp1*temp2);
                ivar[ip][it][io] = ivar[ip][io][it];
            }
        }
    }
}

```

```

    }
    }
    /* output results */
    for(ip=1; ip<=noclas; ++ip) {
        printf(fd2, "      Class number %d      %f      points\n\n\n",
            ip, claent[ip]);
        printf(fd2, "
                                Actual      Simulated\n")
        printf(fd2, "                                Mean      mean\n");
        for(ix=1; ix<=nochan; ++ix) {
            ninc = nocomp * noclas + (ip - 1) * nochan;
            printf(fd2, "      Channel %d ( %6.3f - %6.3f )      %6.3f      %6.3f\n",
                fetvc3[ix-1], lower[ix-1], upper[ix-1], z2[ninc+ix],
                rmean[ip][ix]);
            mobuf[ix-1] = rmean[ip][ix];
        }
        err5 = write(fd3, mobuf, 4*nochan);
        if(err5<0)
            printf("Error occurred writing Mean Vector file (MAIN) \n");
        printf(fd2, "\n\n\n\n\n      Actual Covariance Matrix\n");
        for(no=1; no<=nocomp; ++no) {
            ninc = (ip-1) * nocomp;
            a[no] = z2[ninc+no];
        }
        writmx(a, fetvc3);
        printf(fd2, "\n\n\n\n\n      Simulated Covariance Matrix\n");
        no = 0;
        for(io=1; io<=nochan; ++io) {
            for(in=1; in<=io; ++in) {
                no = no + 1;
                a[no] = ivar[ip][io][in];
                mobuf[no-1] = ivar[ip][io][in];
            }
        }
        writmx(a, fetvc3);
        err5 = write(fd4, mobuf, 4*nocomp);
        if(err5<0)
            printf("Error occurred writing cm file (MAIN) \n");
        printf(fd2, "\n\n\n");
    }
    /* cexit terminates activities on open files and flushes
    * the output buffer. cexit is part of the c library */
    printf("write.c is finished\n");
    printf("\n The simulated data (in PDS format) is at %s \n", pname);
    printf("The lincrinter output file is at %s \n", orname);
    printf("The Mean vector file is at %s \n", mufile);
    printf("The Covariance Matrix file is at %s \n", cmfile);
    pdsoclose(pfd1);
    cexit();
}
/* end of main program */
/*****
/* function to read records 1 thru 5 follows */
readlto5(buf, fetvc3, word, info)
    int *fetvc3, *info;
    char *buf;
    float *word;
{
    extern int noclas, nochan, nocomp, fd1, noents, noline;
    extern float upper[5], lower[5];

```

ORIGINAL PAGE IS  
OF POOR QUALITY



```

int enri, j, k, recnum, statsize;
char tbuf[4];
float ibmdec();
fd1 = open("/dev/rmt0", 0);
if(fd1<0)
    printf("Cannot open 9 trk tape file (READ1T05) \n");
/* skip record 1 */
enri = read(fd1, buf, 1500);
if(enri == -1)
    printf("Error occurred in reading record 1 (READ1T05) \n");
/* read record 2 */
enri = read(fd1, buf, 1500);
if(enri == -1)
    printf("Error occurred reading record 2 (READ1T05) \n");
nochan = buf[19];
nochan = buf[23];
norool = buf[31];
for(j=0, k=33; j<nochan; ++j, k=k+4)
    fctwo3[j] = buf[k];
for(k=32+4*nochan, j=0; j<2*nochan; ++j, k=k+4) {
    tbuf[0] = buf[k];
    tbuf[1] = buf[k+1];
    tbuf[2] = buf[k+2];
    tbuf[3] = buf[k+3];
    if(j<nochan)
        lower[j] = ibmdec(tbuf);
    else upper[j-nochan] = ibmdec(tbuf);
}
/* skip record 3 */
enri = read(fd1, buf, 1500);
if(enri == -1)
    printf("Error occurred reading record 3 (READ1T05) \n");
recnum = buf[11];
while(recnum == 3) {
    enri = read(fd1, buf, 1500);
    if(enri == -1)
        printf("Error occurred reading record three (READ1T05)\n");
    recnum = buf[11];
}
/* Handle record 4 */
statsize = nochan*norool + (nochan*(nochan+1)/2)*norool;
for(k=0, j=16; k<statsize; ++k, j=j+4) {
    tbuf[0] = buf[j];
    tbuf[1] = buf[j+1];
    tbuf[2] = buf[j+2];
    tbuf[3] = buf[j+3];
    word[k] = ibmdec(tbuf);
}
enri = read(fd1, buf, 1500);
if(enri == -1)
    printf("Error occurred reading record 5 (READ1T05) \n");
noents = (buf[18]<<8 | (buf[19] & 0377));
noline = (buf[22]<<8 | (buf[23] & 0377));
info[4] = (buf[38]<<8 | (buf[39] & 0377));
info[5] = (buf[42]<<8 | (buf[43] & 0377));
info[6] = (buf[46]<<8 | (buf[47] & 0377));
info[7] = (buf[50]<<8 | (buf[51] & 0377));
info[8] = (buf[54]<<8 | (buf[55] & 0377));
info[9] = (buf[58]<<8 | (buf[59] & 0377));

```

```

int noolaz, noPool, nochan, fdi, nornts, noline;
float upper[5], lower[5];
float ibmdec(buf)
    char *buf;
{
    int k, l, m, tint;
    char nbuf[2], tchar, temp3, byte[4], sign;
    float word;
    m = 0;
    sign = 0000;
    if(buf[0] < 0)
        sign = 0200;
    tint = (((buf[0] & 0177) - 64) * 4);
    while(buf[1] > 0) {
        buf[1] = ((buf[1] << 1) & 0376);
        ++m;
    }
    k = 8-m;
    nbuf[0] = buf[2];
    nbuf[1] = buf[3];
    for(l=1; l<=k; ++l)
        nbuf[0] = ((nbuf[0] >> 1) & 0177);
        nbuf[1] = ((nbuf[1] >> 1) & 0177);
    for(l=1; l<=m; ++l) {
        buf[2] = ((buf[2] << 1) & 0376);
        buf[3] = ((buf[3] << 1) & 0376);
    }
    buf[1] = (buf[1] | nbuf[0]);
    buf[2] = (buf[2] | nbuf[1]);
    tint = tint-m+128;
    if(tint < 0) tchar = 0000;
    else if(tint > 255) tchar = 0377;
    else tchar = tint;
    buf[1] = (buf[1] & 0177);
    temp3 = tchar << 7;
    temp3 = temp3 & 0200;
    buf[1] = temp3 | buf[1];
    buf[0] = tchar >> 1;
    buf[0] = buf[0] & 0177;
    buf[0] = buf[0] | sign;
    byte[0] = buf[1] & 0377;
    byte[1] = buf[0] & 0377;
    byte[2] = buf[3] & 0377;
    byte[3] = buf[2] & 0377;
    pack(byte, &word);
    return(word);
}
/* the function pack takes the 4 8-bit character bytes
 * that have been rearranged by ibmdec and packs
 * them into a floating point word */
pack(byte, cword)
    char *byte, *cword; {
    int j;
    for(j=0; j<4; ++j)
        cword[j] = byte[j];
}
/*****
/* function to read record & follow. */
read6rec(buf, rntcl);

```

```

int *entols;
char *buf; (
extern int nopts, recnum, fd1;
int j, n, err1, k;
n = 0;
err1=read(fd1, buf, 1500);
if(err1 == -1)
    printf("Error occurred reading record 6\n");
recnum=buf[11];
for(j=20, k=0; k<nopts; k++, j=j+2) (
    entols[n]=(buf[j+1] & 0377);
    ++n;
)
)
int recnum;
/*****
/* Mfsd factors the matrices passed by the main program.
* Mfsd is a direct translation from the Lars
* fortran version of the same name */
mfsd(a)
double *a; (
extern int nochan, ier;
extern float eps;
int kpiu, k, ind, lend, i, lanf, l, lind;
double tol, dsum, dpiu, y, fabs();
double done, snt(), x;
done = 1.0;
ier = 0;
kpiu = 0;
for(k=1; k<nochan; ++k) (
    kpiu = kpiu+k;
    ind = kpiu;
    lend = k-1;
    y = eps*a[kpiu];
    tol = fabs(y);
    for(i=k; i<nochan; ++i) (
        dsum = 0.0;
        if(lend != 0) (
            for(l=1, l<=lend; ++l) (
                lanf = kpiu-l;
                lind = ind-l;
                dsum = (dsum+(a[lanf]*a[lind]));
            )
        )
        dsum = a[lind]-dsum;
        if((i-k) == 0) (
            if(((dsum-tol) <= 0.0) && (dsum > 0.0) && (ier<=0))
                ier = k-1;
            if(((dsum-tol) <= 0.0) && (dsum <= 0.0)) (
                ier = -1;
                printf("Error -1\n");
                exit(1);
            )
            x = dsum;
            dpiu = snt(x);
            a[kpiu] = dpiu;
            dpiu = done/dpiu;
        )
        if((i-k) != 0)
            a[lind] = dsum*dpiu;
    )
)

```

ORIGINAL PAGE IS  
OF POOR QUALITY

```

        ind = ind+1;
    }
}

int ier;
float eps;
/*****
/* the write matrix function writes to the output
* file the lower half of the nochannochan covariance matrix
* which is passed through the parameter a. */
write(a, fctvc3)
    int *fctvc3;
    double *a; {
    extern int fd2, nochan;
    extern float upper[5], lower[5];
    int j, k, m;
    m = 0;
    printf(fd2, "\n Spectral ");
    for(j=0; j<nochan; ++j)
        printf(fd2, "%7.5f - ", lower[j]);
    printf(fd2, "\n Band ");
    for(j=0; j<nochan; ++j)
        printf(fd2, "%7.5f ", upper[j]);
    for(m=0; m<nochan; ++m) {
        printf(fd2, "\n\n %7.5f -\n", lower[j]);
        printf(fd2, " %7.5f ", upper[j]);
        for(k=0; k<m; ++k) {
            printf(fd2, "%7.3f ", a[+m]);
        }
    }
}

int fd2;
/*****
setc(p, s)
char **s; {
    char c, *p1;
    printf(p);
    p1 = s;
    while((c = getch()) != '\n')
        *p1++ = c;
    *p1 = 0;
}

```